```
clear variables

load('lab10_1.mat');

u_id = id.InputData;
y_id = id.OutputData;
u_val = val.InputData;
y_val = val.OutputData;

na = 3*n;
nb = na;

% Initial matrix
P = 100 * eye(na+nb);
% Parameter vector
theta = zeros(na+nb,1);
N = length(y_id);
% Regressor vector
phi = zeros(na+nb,N);

for k = 1 : 1 : N
    for i = 1 : 1 : na
        % adding the elements containing y
        if (k > i)
            phi(i,k) = -1 * y_id(k-i);
        else
            phi(i,k) = 0;
        end
    end
    for i = 1 : 1 : nb
        % adding the elements with u
        if (k > i)
            phi(i+na,k) = u_id(k-i);
        else
            phi(i+na,k) = 0;
        end
    end

    % Find the prediction error
    e = y_id(k) - transpose(phi(:,k))*theta(:,k);
    % Update the inverse
    P = P - (((P*phi(:,k))*transpose(phi(:,k)))*P) / (1 + (transpose(phi(:,k))*P)*phi(:,k));
    % Compute weights
    W = P * phi(:,k);
    % Update parameters
    theta(:,k+1) = theta(:,k) + W*e;

end
% We take theta as the k-th column (theta(k))
 theta = theta(:,end);
% We transpose theta in order to obtain the coefficients for the vectors A and B
 theta = transpose(theta);
 A = [1 theta(1:na)];
```

```
 B = [0 theta(na+1:na+nb)];

% Finally, we are building the model
recmodel = idpoly(A, B, [], [], [], 0, val.Ts);
% Simulating it on the validation data
compare(recmodel,val);
```